UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona

Northeastern
University

Exploring Methods to Enhance Appearance-Based Video Object Tracking using Dynamics Theory

A Master Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

and Northeastern University

By

Marina Alonso Poal

In partial fulfilment of the requirements for the

Master in Advanced Telecommunication Technologies

Advisors

Prof. Octavia Camps

Prof. Xavier Giró

Boston, June 2020

Abstract

The task of Video Object Tracking has for a long time received attention within the field of Computer Vision, and many different approaches have tried to tackle its challenges, being the ones based on appearance and motion some of the most popular ones. The main focus of this thesis is to fuse both strategies in order to exploit their strengths and overcome each other's flaws.

To achieve this goal, we propose a unified framework that combines, in an online manner, an off-the-shelf single-object siamese tracker, which is modified to perform multi-object tracking and to provide more than one detection candidate, with a novel motion module. This module detects when the proposed target position is not dynamically consistent and, if that is the case, predicts an alternative which is used to choose the best among the rest of candidates.

Our approach is evaluated on the challenging Similar Multi-Object Tracking (SMOT) dataset and achieves a relevant precision improvement of the 5% with respect to the baseline. We present an extension to the SMOT dataset, the eSMOT, including more sequences with complex dynamic scenarios, where the performance of our model is excellent, therefore we use its predictions to label the Ground Truth.

Although there is still room for enhancement mainly regarding the efficiency of the approach, this work has served as a relevant proof of concept for the intuitions behind it and consequently, research in this direction will surely continue.

Acknowledgements

First of all, I would like to thank Professor Octavia Camps and Professor Mario Sznaier for inviting and welcoming me to their Laboratory, as well as providing me with supervision and useful advice throughout this thesis.

In addition, I would like to thank Professor Xavier Giró-i-Nieto for his guidance, and acknowledge his constant and genuine efforts on providing his students with the best academic training and opportunities.

I wish to sincerely thank Ponç Palau Puigdevall for always being there, going through hard times with me and celebrating each one of my accomplishments. It has been an absolute pleasure to share this experience with him and I have truly cherished it all.

Finally, I would also like to show my deepest appreciation to my family, for their uninterrupted and unconditional support throughout all my life.

Contents

List of Acronyms

AR	Auto-Regressive
CVPR	Computer Vision and Pattern Recognition
DM	Dynamics Module
GLA	Generalized Linear Assignment
MOT	Multiple Object Tracking
PSD	Positive Semi-Definite
RSL	Robust Systems Laboratory
RoW	Response of candidate Window
SMOT	Similar Multi-Object Tracking
SOT	Single Object Tracking
SVD	Singular Value Decomposition
VOT	Visual Object Tracking
VOS	Visual Object Segmentation

List of Figures

Chapter 1

Introduction

1.1 Context

This thesis has been carried out at the Robust Systems Lab (RSL)[1] at Northeastern University Boston, MA. This laboratory is coordinated jointly by Professor Octavia Camps, whose primary research is related to Computer Vision, and Professor Mario Sznaier, specialized in robust control and optimization. As a result of such a diverse environment, this thesis strategically combines control concepts with vision techniques to solve the task of Visual Object Tracking (VOT).

Although my stay at the RSL has lasted one whole academic year, this thesis has been developed in four months approximately. During the first part of the stage, I was involved in two main projects executed in collaboration with fellow students and PhD candidates at the laboratory. These projects were regarding *Subspace Clustering using a Semi-Algebraic Optimization Approach* and *Video Object Segmentation using Generative Adversarial Networks*.

1.2 Related Work

Object tracking is one of the most relevant tasks in the field of computer vision, and it is used in a wide range of scenarios such as automatic surveillance, vehicle navigation, or video labelling. Given the location of an arbitrary target of interest in the first frame of a video, the aim of VOT is to estimate its position in all the subsequent frames with the best possible accuracy.

Throughout the last decades, many different approaches have been considered to tackle the task of object tracking. A common classification focuses on the features that are used to establish correspondences between the target object and the next frame. These features can be extracted in a handcrafted way (such as Histogram of Oriented Gradients (HOG) or Scale-Invariant Feature Transform (SIFT)) or most recently, using a deep feature extractor such as a Convolutional Neural Network (CNN).

[1]This work was supported in part by NSF grants IIS–1318145, ECCS–1404163, and CMMI–1638234; AFOSR grant FA9550-15-1-0392; and the Alert DHS Center of Excellence under Award Number 2013-ST-061-ED0001.

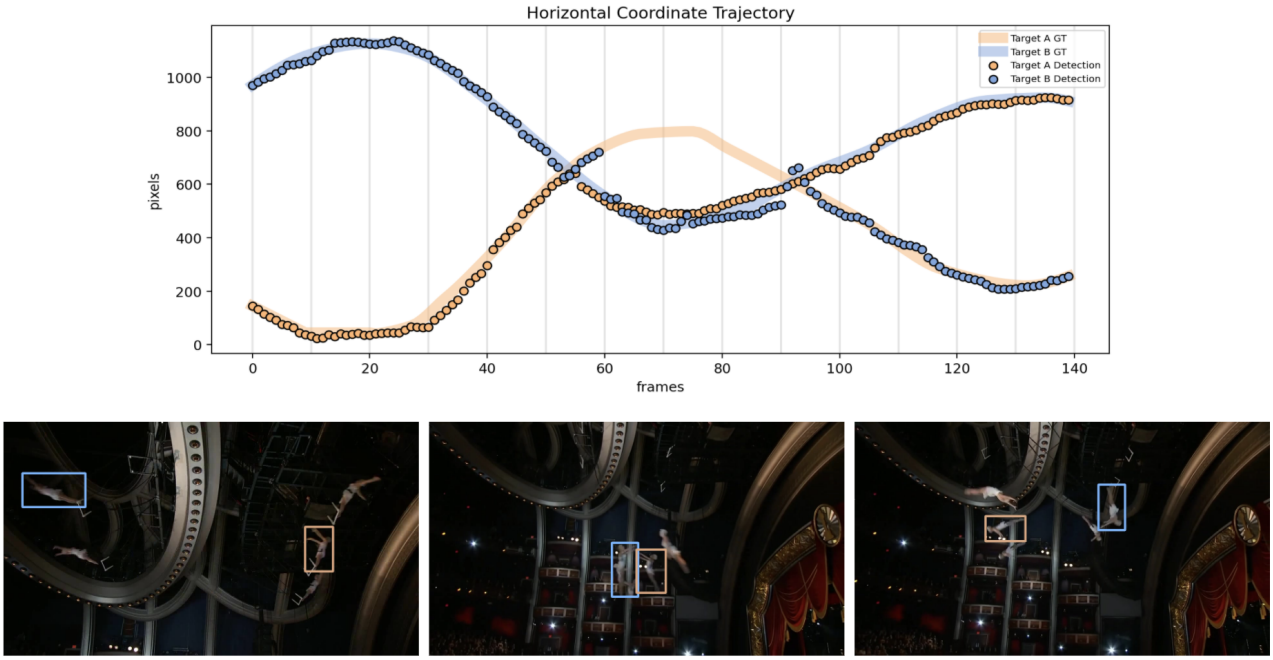

Figure 1.2.1: On the top, trajectory evolution of the horizontal component of the centroids of two different targets with identical appearance. Continuous lines represent the Ground Truth trajectories and points depict an appearance-based tracker detections. Data corresponds to SiamMask performance on sequence *acrobats* from the SMOT dataset. On the bottom, frames 1, 54 and 71 from the sequence, with the corresponding GT bounding boxes for targets A and B.

Independently of how their features are obtained, trackers are usually classified based on what kind of characteristics they use to track, being appearance and motion two of the most popular ones.

On the one hand, appearance-based trackers such as [6] or [10] have gained popularity over the years and give excellent results where the appearance of the targets is discriminative. However, tracking methods that only rely on appearance information struggle in specific scenarios where, for instance, there are multiple objects with similar aspect that act as background distractors. *Figure 1.2.1* exemplifies this phenomenon, where an appearance-based tracker fails to disentangle the trajectories of two objects with identical aspect.

On the other hand, it is also plausible to track exclusively based on dynamics using traditional approaches such as Kalman [11] or particle filters [15] to predict the target location and associate the closest detection to this prediction. However, these approaches must assume a dynamic model *a priori* and often have trouble distinguishing close to each other targets.

Several works [9], [4], have focused on combining both kinds of cues to overcome each other flaw's. In these cases, motion cues provide rich complementary information that can disambiguate the target. While appearance features only encode static information from a single frame, motion features capture the dynamic nature of a scene that is complementary to aspect features.

In [5], which was developed some years ago at RSL by Dicle *et al.*, a computationally efficient algorithm for multi-object tracking by detection was presented. It addressed different challenges

such as appearance similarity among targets or missing data due to targets being out of the field of view, among others. This was accomplished by formulating the problem as a Generalized Linear Assignment (GLA) of tracklets which were incrementally associated into longer trajectories based on their dynamics-based similarity and using efficient algorithms to estimate these similarity measures.

1.3 Contribution

In this thesis we take inspiration from the work proposed by Dicle and combine some of its dynamics theory related techniques with SiamMask [21], a state-of-the-art appearance-based tracker, to try to enhance its performance. In his work, Dicle is interested in distinguishing targets, minimizing minsidentification and recover artificially created missing data, assuming that there is one predecessor for each successor. Divergently, we focus on working online with detections provided by a modified appearance-based tracker and use dynamics cues to correct them, if considered erroneous.

The task we are solving in this thesis is a combination between Single Object Tracking (SOT) and Multiple Object Tracking (MOT). We part from an *off-the-shelf* siamese single-object tracker, where the appearance of the target is known *a priori* and modify it to perform multi-object tracking. However, although each target is tracked at the same time, each of them is treated independently, for it would not be incorrect to say that we are applying a SOT model directly to solve MOT. This usually leads to poor results, as models usually struggle in distinguishing between similar looking objects. Nonetheless, that is not our case, as that is precisely why we add a whole Dynamics Module (DM) to the pipeline. Throughout the course of this thesis, illustrative examples will be given on the task of SOT when explaining the DM but both qualitative and quantitative metrics will be reported on SOT and MOT. It is worth mentioning the fact that, as we have chosen as our starting point a tracker architecture that additionally solves Video Object Segmentation (VOS), we are also able to provide improved qualitative results on this task.

With our approach, we improve the performance of *SiamMask* by a 5% on the Similar Multi-Object Tracking (SMOT) dataset [5]. We also introduce a novel extension to that dataset (eSMOT) that, on the same line, consists of extremely challenging sequences of sports events where multiple objects with similar appearance interact with each other. As our model excelently tracks the target in eSMOT, we have used its predictions to directly label the Ground Truth of the dataset.

The rest of this thesis is organized as follows: in **Chapter 2** we introduce the fundamental theory to understand the intuition behind our presented approach, focusing on presenting siamese trackers and dynamics theory. Next, in **Chapter 3**, we describe the proposed approach in a detailed manner. We extensively cover the modifications made to the baseline tracker, as well as our classification and prediction algorithms in the proposed dynamics module. In **Chapter 4**, we describe the implementation details of our work and present the Similar Multi-Object Tracking (SMOT) dataset. Additionally, the metrics used to evaluate the proposed method are presented and reported. The conclusion and future development proposals for the continuation of this thesis are presented in **Chapter 5**.

Chapter 2

Background

To understand the architectures and methods that will be explained further in this document, it is important to firstly explain the foundations on which they are built. In this section we introduce fundamental theoretical concepts behind siamese networks and dynamics theory and provide references for supplementary information.

2.1 Siamese Trackers

For many years, the most successful approach for tracking an object in a video, where the target is identified solely by a rectangle in the first frame, was to learn a model of the object's appearance in an online fashion using examples extracted from the video itself. However, using this paradigm, only relatively simple models could be learnt [20]. With the adoption of deep convolutional networks, in 2016, Bertinetto *et. al* in [2], presented a novel paradigm for performing object tracking. They introduced *SiamFC*, a simple fully-convolutional siamese network trained end-to-end that operated beyond real-time and achieved *state-of-the-art* performance, making headway for a new line of research regarding the VOT task.

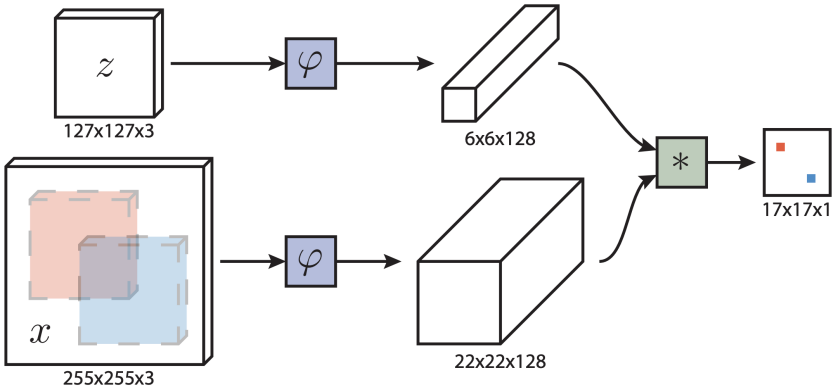

Figure 2.1.1: Fully-convolutional Siamese architecture from *SiamFC*. Image taken from [2]

Siamese networks get their name from the fact that there are two twin neural networks in play that share the parameter space between them, as show in *Figure 2.1.1*. Essentially, siamese networks create a high dimensional embedding representation of their input features. This embedding is trained with a differentiating metric, such as L2 distance or a contrastive loss, that creates a paradigm in which the similarity or dis-similarity between the inputs fed to the twin network can be measured.

Following this reasoning, *SiamFC* compares a **target image** z against a larger **search image** x to obtain a dense response map. In order to obtain this map, the approach uses two siamese convolutional neural netwoks φ that extract appearance-features. The resulting generated map for the target image, is slided over the generated map for the search image and a matrix of similarity measures is used to combine the two matrices into a single scoring matrix. Finally, the region of the search image x corresponding to the maximum value in the scoring matrix, is considered the region in which the target image appears.

2.1.1 SiamMask

In the 2019 edition of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Wang *et. al*, including the original *SiamFC* authors, presented *SiamMask* [21], an enhanced siamese tracker that not only solved the task of Video Object Tracking, but also performed Video Object Segmentation, improving the results of its predecessor *SiamFC* by 5%.

The key improvement that led to these notable results was the incorporation of a new segmentation branch, in addition to the two branches of *SiamFC* and performing depth-wise/channel-wise correlation instead of cross-correlation. Before entering into details, let us introduce the term *Response of a candidate Window* (RoW) defined in [21], which refers to each one of the deep features obtained by the depth-wise correlation between the features of target and search images. In each RoW, we have a deep feature of dimension $1\times1\times256$ encoding how similar the target image and the *n-th* candidate window in the search image are. From each of these deep features or RoWs, SiamMask produces three tensors, each one containing:

- Bounding boxes coordinates (**box** tensor in *Figure 2.1.1*) generated by deep network b_σ. For each RoW, b_σ generates k bounding boxes surrounding the most similar object (appearance-wise) to the target object in that RoW, where k is an hyperparameter indicating how many anchor boxes we use to fit that object. The tensor contains the 4 coordinates needed to represent the bounding box.
- Bounding boxes scores (**score** tensor in *Figure 2.1.1*) generated by deep network s_φ. From each RoW, SiamMask produces $2k$ scores for each of the bounding boxes in box tensor, divided into foreground and background score.
- Deep features needed to produce masks (**mask** tensor in *Figure 2.1.1*), generated by deep network h_ϕ. This tensor contains a $1\times1\times(63\times63)$ feature vector for each RoW, needed to generate the mask with a subnetwork.

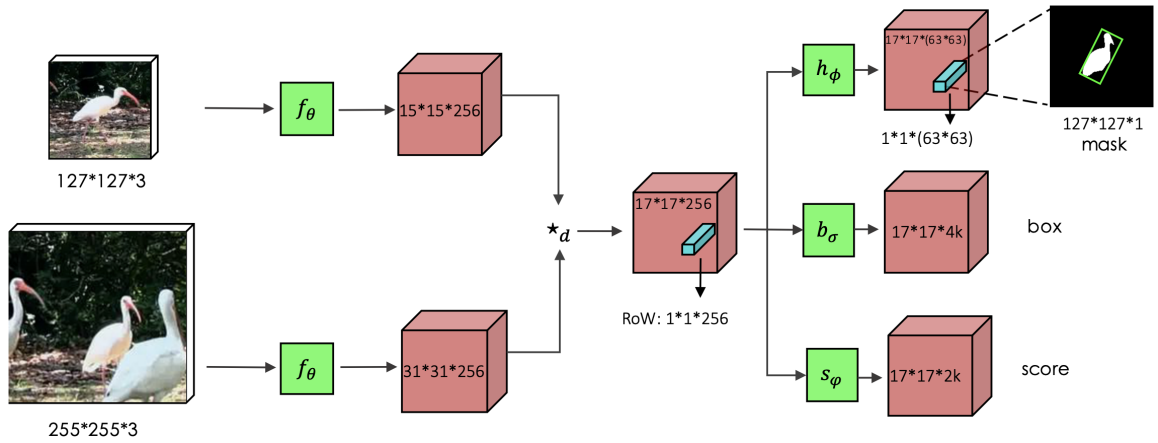

Figure 2.1.2: Schematic illustration of *SiamMask*'s three-branch architecture. Image taken from [21]

Finally, SiamMask uses the RoW with the highest score in *score* tensor, which depends on the *box* tensor, to generate a mask with the RoW coming from the *mask* branch.

2.2 Dynamics Theory

A fundamental part of the approach presented in this thesis consists of detecting when a bounding box proposed by the siamese tracker is consistent with the past trajectory of the target object. In this subsection we present the main dynamics theory concepts upon which our work is based.

2.2.1 Hankel and Gram Matrices Representation

Let us first introduce the concept of *tracklet*. A tracklet α consists of an ordered sequence of measurements y_k, $s \leq k \leq e$, where s and e are the starting and ending times, respectively. The underlying dynamics of the tracklet can be represented using an Auto-Regressive (AR) model, with the following form, for a high-enough value of n:

$$y_k = \sum_{i=1}^{n} a_i \, y_{k-i} \quad k \geq s + n \tag{2.1}$$

The order n of the model measures the *complexity* of the underlying dynamics and, in the absence of noise, it can be estimated as the rank of the Hankel matrix corresponding to that sequence. The Hankel matrix is a matrix commonly used in linear algebra, in which each ascending skew-diagonal from left to right is constant. In the context of measuring the complexity of a sequence, we define the Hankel matrix $\mathbf{H}_\alpha^{(m)}$ of tracklet α with $m \geq n$ columns matrix as follows:

$$\boldsymbol{H}_{\alpha}^{(m)} = \begin{bmatrix} y_s & y_{s+1} & \cdots & y_{s+m-1} \\ y_{s+1} & y_{s+2} & \cdots & y_{s+m} \\ \vdots & \vdots & \vdots & \vdots \\ y_{e-m+1} & y_{e-m} & \cdots & y_e \end{bmatrix} \tag{2.2}$$

Additionally, as it is mentioned in [14], the subspace spanned by the columns of the Hankel matrix characterizes the tracklet's dynamics and is invariant to both initial condition and affine viewpoint changes. Thus, in principle, Hankel matrices could be related by simply comparing the angles between their subspaces spanned by the respective columns. However, under the presence of noise, the Hankel matrix tends to be full rank, and the angle between subspaces becomes zero, making the comparison unreliable.

To avoid this issue, in [22], Zhang *et. al* proposed to work with Gram matrices followed by a Frobenius normalization, as depicted in *Equation 2.3*. Gram matrices inherit the rank and invariance properties of the associated Hankel matrices. However, in contrast to the later, Gramm matrices are confined to the Positive Semi Definite (PSD) manifold, which will be useful for this purpose.

$$\hat{\boldsymbol{G}}_{\alpha}^{(m)} = \frac{\boldsymbol{H}_{\alpha}^{(m)} \boldsymbol{H}_{\alpha}^{(m)T}}{||\boldsymbol{H}_{\alpha}^{(m)} \boldsymbol{H}_{\alpha}^{(m)T}||_F} \tag{2.3}$$

2.2.2 Comparing Temporal Sequences

In order to compare two temporal sequences taking into account its dynamic's complexity, [22] suggests to use non-Euclidean similarity functions on the PD manifold, which capture better the underlying geometry than directly comparing the sequences or their Hankel matrices.

In this thesis, we use two commonly used distance-like functions on the PD Riemannian manifold [8], which are the Jensen Bregman Log-det Divergence (JBLD), also known as Stein Divergence, and the The KL-Divergence Metric (KLDM), which is also referred to as Jeffrey Divergence. Their mathematical expressions are articulated below. Computing one of these metrics between a set of PD matrices, such as the previously mentioned Gramm matrix, the dynamic similarity between two sequences can be efficiently and accurately measured.

$$\delta_{jbld}^2(\boldsymbol{X}, \boldsymbol{Y}) = log \left| \frac{\boldsymbol{X} + \boldsymbol{Y}}{2} \right| - \frac{1}{2} log|\boldsymbol{X}\boldsymbol{Y}| \tag{2.4}$$

$$\delta_{jkl}^2(\boldsymbol{X}, \boldsymbol{Y}) = \frac{1}{2} Tr(\boldsymbol{X^{-1}Y} + \boldsymbol{Y^{-1}X} - 2\boldsymbol{I}) \tag{2.5}$$

2.2.3 Hankel Total Least Squares

Dicle *et. al* proposed in [5] an algorithm to estimate the rank of an incomplete Hankel matrix corrupted with additive noise. The algorithm was based on two simple modifications of the Hankel Total Least Squares (HTLS) algorithm [16] which allowed them to handle missing data and to estimate rank. The two modifications to the original algorithm are the following:

- Introduce an "indicator" binary vector to flag missing data and allow its recovery while performing inpainting to stitch tracklets with gaps.
- Run this algorithm, iteratively, for increasing rank values to find the optimal rank n of the sequence.

As this algorithm, known as Iterative HTLS has been used this thesis and we have provided a novel *PyTorch* implementation for it, we proceed to explain it in detail.

Let us first formulate the problem: consider the previously introduced tracklet $\alpha = [y_s, ..., y_e]$, of length l and known dynamics complexity $n < l$. Let $\hat{\alpha} = [\hat{y_s}, ..., \hat{y_e}]$ and $\eta = [\eta_s, ..., \eta_e]$ be respectively, the noiseless estimated measurements and the noise. Additionally, let ω be the l-length binary vector where $0s$ indicate missing measurements.

The solution presented is for scalar measurements for the sake of simplicity. However, generalization to the multi-dimensional case is straight forward. Also for simplicity of notation, let $[\boldsymbol{A}|b] = [\boldsymbol{H}_{\alpha}^{(n)}|b]$ and $[\boldsymbol{E}|f] = [-\boldsymbol{H}_{\eta}^{(n)}|f]$, where $b = [y_{s+n}^T, ..., y_e^T]^T$ and $f = [\eta_{s+n}^T, ..., \eta_e^T]^T$. From 2.1 and 2.2, it can be proven that there exists a vector x such that: $(\boldsymbol{A} + \boldsymbol{E})x = (b + f)$ and α can be estimated by solving the following modified Total Least Squares (TLS) problem:

$$\min_{x, \boldsymbol{E}, f} ||\boldsymbol{\Omega} \circ [\boldsymbol{E}|f]||_F^2 \qquad st.\ (\boldsymbol{A} + \boldsymbol{E})x = b + f \tag{2.6}$$

where $\circ$ is the Hadamard product and $\boldsymbol{\Omega} = \boldsymbol{H}_{\omega}^{(n+1)}$ is introduced to recover the missing data. It is worth mentioning that the general TLS problem without missing data has a closed form solution for general matrices which can be found by computing the Singular Value Decompositions (SVD) of $[\boldsymbol{A}|b]$. However, adding the structural constraints precludes a close form solution since SVD does not preserve the Hankel structure. Thus, Dicle *et. al* solve the HTLS problem by using Newton's method which converges in a few iterations. Since $[\boldsymbol{E}|f]$ and $[\boldsymbol{A}|b]$ are Hankel matrices with constant off-diagonals, Equation 2.6 can be rewritten as:

$$\min_{\eta, x} ||\boldsymbol{W}\boldsymbol{D}\eta||_2^2 \qquad st.\ r(\eta, x) = b + f - (\boldsymbol{A} + \boldsymbol{E})x = 0 \tag{2.7}$$

where $\boldsymbol{D}$ is a diagonal matrix with the number of times each η_i appears in the Hankel matrix $\boldsymbol{H}_{\eta}^{(n+1)}$ and $\boldsymbol{W} = diag(\omega)$.

Following the approach introduced in [16], the constraint can be combined with the minimization problem as shown in *Equation 2.8*, where π is a large penalty constant.

$$\min_{x,\boldsymbol{E},f} \left\| \begin{pmatrix} \pi r(\eta, x) \\ \boldsymbol{W}\boldsymbol{D}\eta \end{pmatrix} \right\|_2^2 \tag{2.8}$$

Next, $r(\eta, x)$ can be rewritten as $r(\eta, x) = b + \boldsymbol{P_1}\eta - (\boldsymbol{A} + \boldsymbol{E})x$, where $\boldsymbol{P_1} = [\boldsymbol{0}_m \ \boldsymbol{I}_m]$ and $m = l - n$. Linearizing $r(\eta, x)$ results in $r(\eta + \delta\eta, x + \delta x) \approx r(\eta, x) + \boldsymbol{P}_1\delta\eta - (\boldsymbol{A} + \boldsymbol{E})\delta x - \delta\boldsymbol{E}x$. Finally, $\boldsymbol{X} \in \mathbb{R}^{m \times (m+n-1)}$ is defined such that $\boldsymbol{E}x = \boldsymbol{X}\boldsymbol{P_0}\eta$, where $\boldsymbol{P_0} = [\boldsymbol{I}_{(m+n-1)\times(m+n-1)} \ \boldsymbol{0}_{(m+n-1)\times 1}]$. Combining this equation with $r(\eta + \delta\eta, x + \delta x)$, *Equation 2.8* can be rewritten as *Equation 2.9*, which can be solved by least squares.

$$\min_{x,\boldsymbol{E},f} \left\| \begin{pmatrix} \pi(\boldsymbol{P_1} - \boldsymbol{X}\boldsymbol{P_0}) & -\pi(\boldsymbol{A} + \boldsymbol{E}) \\ \boldsymbol{W}\boldsymbol{D} & 0 \end{pmatrix} \begin{pmatrix} \delta\eta \\ \delta x \end{pmatrix} + \begin{pmatrix} \pi r \\ \boldsymbol{W}\boldsymbol{D}\eta \end{pmatrix} \right\|_2^2 \tag{2.9}$$

The procedure of computing HTLS with missing data is detailed in *Algorithm 1*, and its iterable extension is written in *Algorithm 2*. We have made slight modifications in the writing of the algorithms, in order to match with the terminology we use in *Chapter 3* and adopt some novel *PyTorch* implementation details.

Algorithm 1: Hankel Total Least Squares with missing data

Input: α sequence of length l, ω sampling sequence of length l, n desired rank.

Output: $\hat{\alpha}$ inpainted and cleaned sequence, η noise, x AR coefficients.

Form $[\boldsymbol{A}|b]_{(l-n+1)x(n+1)}$

Solve $min_x \|\boldsymbol{A}x - b\|_2^2$

Form $\boldsymbol{P_1}$ and $\boldsymbol{W}\boldsymbol{D}$ from ω

$\eta \leftarrow 0$

while $\left\| \begin{pmatrix} \delta\eta \\ \delta x \end{pmatrix} \right\| > \theta$ **do**

- Form $\boldsymbol{X}\boldsymbol{P_0}$ from x
- Form $[\boldsymbol{E}|f]_{(l-n+1)x(n+1)}$
- Compute $r \leftarrow b + f - (\boldsymbol{A} + \boldsymbol{E})x$
- Form $M \leftarrow \begin{pmatrix} \pi(\boldsymbol{P_1} - \boldsymbol{X}\boldsymbol{P_0}) & -\pi(\boldsymbol{A}+\boldsymbol{E}) \\ \boldsymbol{W}\boldsymbol{D} & 0 \end{pmatrix}$
- Solve $\min_{\delta\eta,\delta x} \left\| M \begin{pmatrix} \delta\eta \\ \delta x \end{pmatrix} + \begin{pmatrix} \pi r \\ \boldsymbol{W}\boldsymbol{D}\eta \end{pmatrix} \right\|_2^2$
- Update $x \leftarrow x + \delta x$
- Update $\eta \leftarrow \eta + \delta\eta$

end

Algorithm 2: Iterative Hankel Total List Squares

Input: α sequence of length l, ω sampling sequence of length l, η_{max} maximum average error.

Output: $\hat{\alpha}$ inpainted and cleaned sequence, n estimated rank for the sequence, x AR coefficients.

$n \leftarrow 0$

$\mu_\eta \leftarrow$ huge

Form $\boldsymbol{\Omega}_{\boldsymbol{(l-n)x(n+1)}} = \boldsymbol{H}_\omega^{(n+1)}$

while $\mu_\eta > \eta_{max}$ **do**

  $n \leftarrow n + 1$

  $(\hat{\alpha},\ \eta,\ x) \leftarrow$ Solve HTLS problem given $(\alpha,\ n)$

  Form $[\boldsymbol{E}|f]_{(l-n)x(n+1)} \leftarrow \mathcal{H}(\eta)$

  Compute average error $\mu_\eta \leftarrow \frac{||\boldsymbol{\Omega} \circ [\boldsymbol{E}|f]||_F}{||\boldsymbol{\Omega}||_1}$

end

Chapter 3

Methodology

In this section we formalize the problem and describe the proposed approach in a detailed manner. We extensively cover the modifications made to the baseline tracker, as well as our classification and prediction algorithms in the proposed Dynamics Module.

3.1 Pipeline

Let us first illustrate the temporal high-level pipeline of a vanilla visual object tracker. Given a file containing the initialization information (the frame numbers in which the target is initialized and its bounding box size and position), and a set of frames, a tracker iterates over time and distinguishes between two cases: If the object being tracked is initialized in that frame, the tracker updates its template (or target image) and from that frame on, it will search for that specific template in a search area of the image, until it is re-initialized again. If the object is not reinitialized at that frame, the tracker will find, in our case using a siamese architecture as explained in *Chapter 2*, the position where the template appears in the search area given by the previous frame detection. This simple architecture is illustrated in *Figure 3.1.1*.

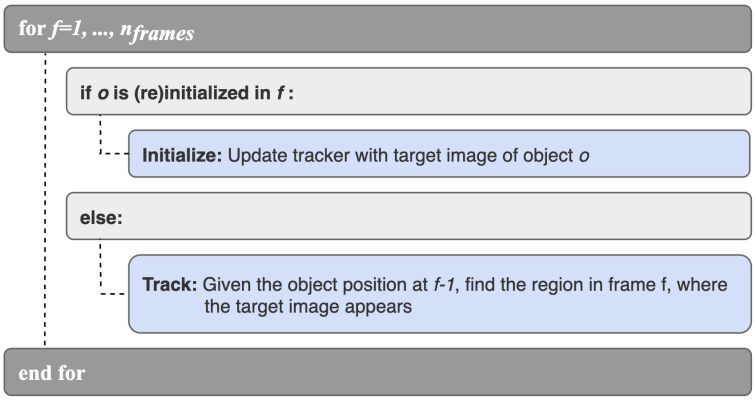

Figure 3.1.1: High-level schematic illustration of the temporal behaviour of a vanilla visual object tracker.

In order to improve the performance of the tracker introducing motion features, we make several modifications to the previous architecture, and most importantly, we add a novel Dynamics Module. We will first explain the pipeline of our approach and in the following subsections each modification will be described in detail.

In this new approach, which is depicted in *Figure 3.1.2*, we are able to track different objects at the same time, iterating over each object at each frame and treating them independently. Every time that an object is initialized, in this architecture we also initialize a Dynamics Module. When an object is not being initialized at a given frame, the siamese network will return the bounding box where it considers that the template appears in the search area given by the previous frame detection. Additionally, we force the tracker to return N additional bounding boxes candidates, among which we will be able to find the most dynamically consistent, if the one proposed by the network is considered erroneous.

In order to do so, at each frame and for every object, the Dynamics Module is updated with the proposed target position by the network and, if a sufficient number of frames (T_0) have occurred since

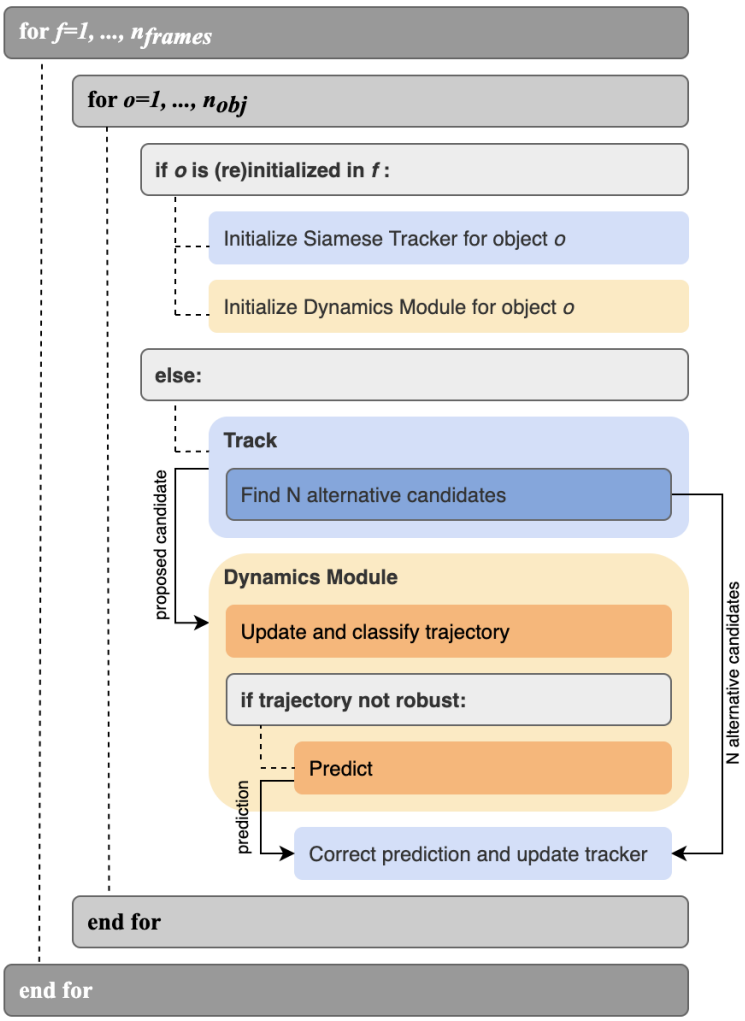

Figure 3.1.2: High-level schematic illustration of our proposed architecture.

the object was initialized, it determines if the proposed position is dynamically coherent with the object trajectory. If the new sample represents an inconsistent motion, the Dynamics Module predicts and outputs a robust new measurement.

Finally, if a prediction has been made, the tracker is modified to filter some of the N provided candidate bounding boxes, taking into consideration the size and position of the target at the previous frame, and among the resulting candidates, it chooses the one whose centroid is closer to the predicted position. The size and centroid position of this bounding box is considered to be the correct location of the object at that given frame and that information is updated in the tracker, so in the next frame, it will look for the template image around that position.

3.2 Tracker Modifications

Three main modifications have been done to the original *off-the-shelf* tracker, excluding the incorporation of the Dynamics Module. These three modifications are the following: converting architecture to perform multi-object tracking, extracting N alternative candidates when tracking and finding the best alternative candidate given the Dynamics Module's prediction. Each of these modifications will be thoroughly explained in the upcoming subsections.

3.2.1 Multi-Object Tracking

Until recently, it was not common to see in the literature siamese trackers solving the task of multi-object tracking, because they were considered to be inefficient for this task. However, in the recent years, some works such as [3] or [19] have appeared, where a two stage detect-and-track framework is proposed, which includes in its traditional siamese architecture, two additional branches, one to estimate the object motion and one re-identification branch that re-activates the previously terminated tracks when they re-emerge.

To test our algorithm for MOT, we opted to perform a not efficient yet effective modification, that consisted on, as it can be seen in *Figure 3.1.2*, iterating over each object at each frame and storing all of its updated and required data in a dictionary.

3.2.2 Alternative Candidates Extraction

When tracking similar objects in video sequences, it is likely that more than one object is present in the search area of the target image. Therefore, we thought about exploiting the cross-correlation product present in siamese trackers to find the position of similar objects around the target object. Concretely, in *SiamMask* we use the *score* and *box* branches to find N alternative candidates. The details of how this procedure is carried out are defined below:

Figure 3.2.1: Representation of $(\Delta x, \Delta y)$ in an image. We can see the scores associated to the anchor box with highest score in each $(\Delta x, \Delta y)$, matching the location of similar objects appearance-wise.

Recalling the structure shown in 2.1.2, let us define the pair $(\Delta x, \Delta y)$ as the position of the RoW in the (17×17) grid resulting from the depth-wise cross-correlation product. In each $(\Delta x, \Delta y)$ RoW, we obtain a similarity feature of dimension (1×1×256), indicating *how similar is the target object in relation to the area of the target image corresponding to the receptive field of that particular* $(\Delta x, \Delta y)$ *RoW.* However, although the deep features of *SiamMask* contain a similarity measure preserving the spatial location of the objects, its dimension of (1×1×256) makes them difficult to interpret. On the other hand, in *SiamMask*'s branches we have interpretable information that we can use to find the N most similar targets. Concretely, we propose using the information in *score* and *box* branches.

Score branch outputs a (17×17×2k) tensor containing the background/foreground scores of k anchor boxes whose coordinates are in *box* branch. Therefore, **we take the N maximum boxes with the highest score comimg from different** $(\Delta x, \Delta y)$. It is important that we remark this point because choosing different deltas allows us to look around the object of interest. In *Figure 3.2.1* we display the highest score of the box of each delta, matching the location of similar objects around the target object. When having selected N highest deltas corresponding to the highest N scores, we obtain the bounding box coordinates from the *box branch.*

3.2.3 Finding the Best Candidate

In our first attempt to combine the modified tracker with the motion module, we straightforwardly updated the tracker with the bounding box corresponding to the predicted centroid and the size proposed by the tracker. However, this approach was grounded in the following two incorrect assumptions:

- The centroid position proposed by the dynamics module is perfectly predicted. As it will be explained later in this chapter, the signal upon which the prediction is based is quite noisy, as it comes from real detections. Therefore, the provided prediction is far from exact, although it is accurate in giving the direction and the magnitude of the object's motion.

- The bounding box size proposed by the tracker is always correct. Most of the times, when the Dynamics Module clasifies the incoming sample as dynamically inconsistent, it is because the tracker has made a misdetection, changing the target, merging objects, etc., which leads to incoherent bounding box sizes.

Due to this two incorrect assumptions, this first approach resulted in a poor performance, as it normally got stuck in some space of the image where there was not even an object. To avoid this issue, we decided to find the most dynamically-consistent bounding box candidate provided by the tracker, using the prediction output by the dynamics module. This process consisted of the two following steps illustrated in *Figure 3.2.2*, although perhaps a little overwhelming.

- **Filtering the candidates:** Sometimes, a large number of extracted candidates are requiered for the tracker to output the region corresponding to the correct match. However, in those cases, the network returns bounding boxes with extreme sizes and even in extreme locations. To prevent our architecture to be distracted by these options, we propose a simple heuristic to filter the candidates: if their intersection over union with the bounding box at the previous frame is smaller than th_{iou} or the size of the candidate is much smaller (controlled by th_{sz}), we discard the candidate. These two thresholds are treated as hyperparameters.
- **Selecting the best candidate:** Once we have the predicted position by the Dynamics Module and the filtered candidate bounding box, we select the best closest candidate by computing the Euclidian distance between the prediction and the centroids of each bounding box.

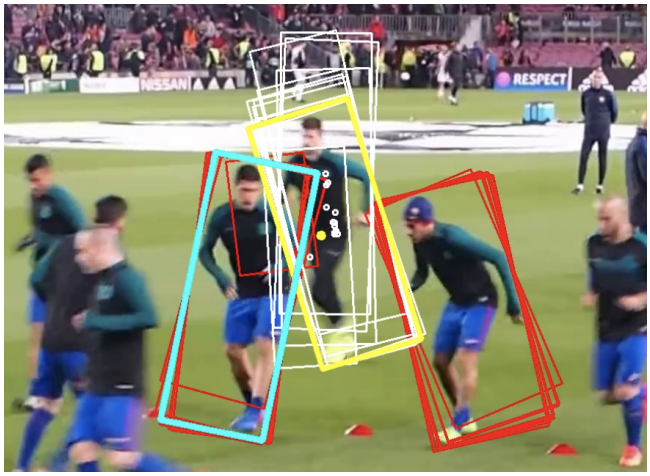

Figure 3.2.2: Example of incorrect target proposal by the network and corresponding correction by our architecture. The object being tracked is the player in the middle, however, the siamese tracker incorrectly proposes the cyan bounding box. We have forced the tracker to output N alternatives, corresponding to the white and red boxes in the image. The red ones are the discarded candidates because they are either too small with respect to the previous detection or do not intersect sufficiently with it. The yellow dot corresponds to the predicted position by the DM, and the white circles are the centroids of the not filtered candidates. Finally, our approach proposes the yellow box as the correct detection, for its centroid has the smallest Euclidean distance to the predicted position.

3.3 Dynamics Module

Our proposed Dynamics Module consists of four sequential parts, which are updating its buffers with the incoming data, studying the dynamic impact that the new measure represents, determining whether this impact is consistent and, if it is not, proposing an alternative. In this section, we cover the main parts of this procedure and describe our proposed algorithms, as well as explain some variations on the approach and provide descriptive figures to illustrate the module's behaviour.

3.3.1 Classification

The input to our classification algorithm mainly consists on a buffer that stores all the tracked centroid positions at each frame. From this buffer, at each time step f and for each coordinate d separately, we extract two signals: $u_k^{(1)}$, which includes the last T_0 positions of the target's centroid at that location and $u_k^{(2)}$, which is the same as the first signal, but with the additional incoming sample. Consequently, the lengths of these sequences are $l^{(1)} = T_0$ and $l^{(2)} = T_0 + 1$.

Based on the concepts detailed in *Chapter 2*, we propose comparing temporal sequences by using a distance-like function on the PD manifold to compare their associated Gram matrices. In this case, as our signals come from real detections from the tracker, we assume that they are corrupted with noise. We make explicit this model in *Equation 3.1* and *Equation 3.2*, where $y_k^{(i)}$ consists of the ideal signal and $\eta_k^{(1)}$ is the additive noise.

$$u_k^{(1)} = y_k^{(1)} + \eta_k^{(1)} \qquad k = f - T_0 - 1, ..., f - 1 \tag{3.1}$$

$$u_k^{(2)} = y_k^{(2)} + \eta_k^{(2)} \qquad k = f - T_0 - 1, ..., f \tag{3.2}$$

In order to compare both signals, we use Dicle's Iterative Hankel Total Least Squares (IHTLS) algorithm, reviewed in *Chapter 2*, to produce $\hat{y}_k^{(1)}$, an estimate of the noiseless first signal, as well as to estimate the noise, $\hat{\eta}_k^{(1)}$ and the order of the sequence n. For the second signal, we use HTLS to find $\hat{y}_k^{(2)}$ and $\hat{\eta}_k^{(2)}$ given the underlying order of the first sequence.

The respective Hankel matrix for each signal is computed, with the number of columns $m = n+1$, resulting in the following structure:

$$\boldsymbol{H}^{(m)}_{\hat{y}_k^{(i)}} = \begin{bmatrix} \hat{y}_1^{(i)} & \hat{y}_2^{(i)} & \cdots & \hat{y}_m^{(i)} \\ \hat{y}_2^{(i)} & \hat{y}_3^{(i)} & \cdots & \hat{y}_{1+m}^{(i)} \\ \vdots & \vdots & \vdots & \vdots \\ \hat{y}_{l^{(i)}-m+1}^{(i)} & \hat{y}_{l^{(i)}-m}^{(i)} & \cdots & \hat{y}_{l^{(i)}}^{(i)} \end{bmatrix} \tag{3.3}$$

As our sequence $y_k^{(i)}$ is an estimation of a noiseless signal, not the signal itself, it will most likely include some residual noise, and its corresponding Gram matrix likely to be full-rank. To avoid this issue, we introduce a regularization factor ϵ, that is commensurate with the estimated noise, in the computation of the Gram matrix, using the following approximation, where n_r is the number of rows of the Hankel matrix and ϵ is our estimated regularization factor.

$$\hat{\boldsymbol{G}}^{(m)}_{\hat{y}_k^{(i)}} = \frac{\boldsymbol{H}^{(m)}_{\hat{y}_k^{(i)}} \boldsymbol{H}^{(m)T}_{\hat{y}_k^{(i)}} + n_r \epsilon \boldsymbol{I}}{\left\| \boldsymbol{H}^{(m)}_{\hat{y}_k^{(i)}} \boldsymbol{H}^{(m)T}_{\hat{y}_k^{(i)}} + n_r \epsilon \boldsymbol{I} \right\|_F} \tag{3.4}$$

The resulting two Gram matrices are compared using the Jensen Bregman Log-det Divergence (JBLD) distance as shown in *Equation 3.5*. As the number of columns in the Hankel matrix has been fixed, the corresponding Gram matrices from both sequences will always have the same shape, even if the sequences have different lengths. Ideally, if the two sequences had the same underlying dynamics, the JBLD distance between them would be zero and otherwise, it will be infinite. For simplicity of notation, we redefine $\hat{\boldsymbol{G}}^{(m)}_{\hat{y}_k^{(1)}}$ as $\boldsymbol{X}$ and $\hat{\boldsymbol{G}}^{(m)}_{\hat{y}_k^{(2)}}$ as $\boldsymbol{Y}$.

$$\delta^2_{jbld}(\boldsymbol{X}, \boldsymbol{Y}) = log \left| \frac{\boldsymbol{X} + \boldsymbol{Y}}{2} \right| - \frac{1}{2} log |\boldsymbol{X}\boldsymbol{Y}| \tag{3.5}$$

To get another measure to base our classification decision on, $\hat{\eta}$ is defined as the difference between the Frobenius norms of both noises ($\hat{\eta}_k^{(1)}$ and $\hat{\eta}_k^{(2)}$), that is, the additional noise that had to be added to the new sequence due to the new sample, to continue having the same rank.

Finally, we use a simple heuristic to determine whether a prediction must be made or not. We empirically define a set of thresholds, that will regulate the areas where the signals will be considered consistent. It is worth noting that we incorporate to our decision algorithm the score (or confidence) with which the tracker has made its detection, for it gives the Dynamic Module a hint on the correctness of its proposal.

We detail this procedure of classifying new incoming measurements between consistent or inconsistent with the target's previous trajectory in *Algorithm 3*. It is worth mentioning that, for the sake of simplicity, we have detailed the approach for one-dimensional sequences, however the conversion to multiple dimensions is straight forward and it is supported by our implementation.

Algorithm 3: Classification algorithm at time $f > T_0$

Input: $\boldsymbol{u}_k$ updated buffer containing centroid positions along time of size: $(f,\ 2)$, T_0 window length, η_{clas}^{max} maximum average error, c tracker confidence, th_{jbld} and th_{jbld}^{max} thresholds for JBLD distance, th_η threshold for HTLS noise, th_{score} and th_{score}^{min} thresholds for score confidence.

Output: $\boldsymbol{p}$ boolean for each coordinate (x, y) indicating if prediction is needed.

$\boldsymbol{p} \leftarrow$ [False, False];

for *d in range (x, y)* **do**

$u_k^{(1)} \leftarrow\ u[f - T_0 - 1 : f - 1,\ d]:$ the current sample is **not** included

$(\hat{y}_k^{(1)},\ \hat{\eta}_k^{(1)},\ n) \leftarrow$ solve **Iterative** HTLS given $(u_k^{(1)},\ \eta_{clas}^{max})$

$u_k^{(2)} \leftarrow\ u[f - T_0 - 1 : f,\ d]:$ the current sample **is** included

$(\hat{y}_k^{(2)},\ \hat{\eta}_k^{(2)}) \leftarrow$ solve HTLS given $(u_k^{(2)},\ n)$

$d \leftarrow$ compare dynamics of $\hat{y}_k^{(1)}$ and $\hat{y}_k^{(2)}$ given n and $\epsilon \sim \eta$

$\eta \leftarrow ||\hat{\eta}_k^{(2)}||_F - ||\hat{\eta}_k^{(1)}||_F$

if $(d \geq th_{jbld}$ **and** $\eta \geq th_\eta$ **and** $c \leq th_{score})$ **or** $(d \geq th_{jbld}^{max})$ **or** $(c \leq th_{score}^{min})$ **then**

$p[d] \leftarrow$ True

end

end

3.3.2 Prediction

Once it has been determined that a new incoming sample provided by the tracker is not dynamically consistent with the previous trajectory, a prediction is performed to guide the tracker towards the correct direction. Two strategies have been used in this work to predict a new sample based on the past trajectory, but both are based on the following first step.

In the classification algorithm, an iterative HTLS step estimated the order of the past sequence, given a maximum noise level η_{clas}^{max} and in the prediction step we are interested in knowing the order of the sequence to be able to correctly forecast a new measure. When classifying however, we are interested in capturing the complexity of the underlying dynamics in an extremely precise manner, therefore we use a small value of η_{clas}^{max}, which leads to higher rank estimations. A phenomenon that could occur, is that the estimated order of the sequence was overfitted due to severe noise presence, and this would lead to extreme incorrect predictions.

Therefore, if the classification algorithm determines that the new measure is not consistent, the Iterative HTLS is again applied, but this time with a higher noise boundary, η_{pred}^{max}, so lower orders are estimated, making the predictions, based on resulting $\hat{y}_k^{(1)}$, more robust.

Tracklet Inpainting

The first strategy used in this algorithm to provide new dynamically consistent measures leverages the computations performed when solving the Iterative Hankel Total Least Squares. In this case, we introduce to the algorithm an "indicator" binary vector flagging the position $T_0 + 1$ of sequence u_{inc} as missing data and allow its recovery while performing inpainting to stitch tracklets with gaps (see *Chapter 2*. This procedure is equivalent to combining the AR coefficients provided by HTLS with the last n components of u_{past}.

State Space Estimation

State Space (SS) is a generic and flexible form of encapsulating many of the most popular linear time series models, allowing estimation with missing observations, forecasting, and so on. A general state space model is of the form:

$$u_k = \boldsymbol{Z}_k y_k + d_k + \epsilon_k \tag{3.6}$$

$$y_{k+1} = \boldsymbol{T}_{kk} + c_k + \boldsymbol{R}_k \eta_k \tag{3.7}$$

where y_k refers to the observation vector at time k, $_k$ refers to the (unobserved) state vector at time k, and the rest are defined as irregular components. For this thesis, as we are working with AR models, we could also rewrite them as state spaces representations. If the output of the IHSTL would have estimated the order of our sequence u as $n = 2$, we could take the expression of *Equation 2.1*, rewrite it for an $AR(2)$ with noise and express it as a SS, as in *Equations 3.6 and 3.7*.

$$u_k = \alpha_1 u_{k-1} + \alpha_2 u_{k-2} + \epsilon_k \tag{3.8}$$

$$u_k = \begin{bmatrix} 1 & 0 \end{bmatrix} y_k; \quad y_{k+1} = \begin{bmatrix} \alpha_1 & \alpha_2 \\ 0 & 0 \end{bmatrix} y_k + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \eta_k \tag{3.9}$$

For our work, we use *Python*'s library **statsmodels** [18] to construct a SS model instance with our data, estimate the model's parameters α_i and ϵ^2_{var} by maximum likelihood and forecast one sample. Explaining the theory behind the estimation of these parameters is out of the scope of our work, so it will not be discussed here. If interested, please refer to [7] for more information.

3.3.3 Variations

An important part of this thesis consisted on studying the impact that different variations of the algorithm had on the results. Some small variations had a significant effect on the performance, but helped fine-tune the architecture to work on videos of different nature. Thus, these little modifications will be further analyzed on *Chapter 4*. However, some other configurations were discarded at an early stage as they induced a poor performance, but we feel it is important to briefly discuss some of them.

- **Working with diverse signals:** In the final configuration, we solely rely on the position of the centroid for each coordinate. Nonetheless, we studied working also with the position of the corners of the bounding box, which was abandoned because they were extremely noisy. Another signal to be taken into consideration was the velocity of the object (difference between positions), but its underlying dynamics were not as descriptive as the ones from the position.

- **Combining coordinates:** Once it was decided that the signal to be treated was going to be the centroid position, we studied whether we should treat each of its coordinates jointly or separately. On the one hand, working with separate coordinates is useful to decouple the misdetected component, however, as both coordinates correspond to the same bounding box, its contribution is not uncorrelated. After trying different configurations, we decided that the best option was to determine the dynamic consistency of each coordinate separately but, if either of them resulted inconsistent, a prediction was always made for both components.

- **Operating over a larger time horizon:** When studying the performance of the DM *off-line*, that is, with pre-loaded tracker data upon which no modification can be done, we studied the effect of working over more than one future frame. However, doing so did not show any significant improvement and it resulted cumbersome to determine which was exactly the sample that corrupted the trajectory's dynamics. Furthermore, in the final architecture, where tracker and DM are combined, operating only over one time step is extremely more efficient.

Figure 3.3.1, helps understand how the final architecture of the Dynamics Module works. The sequence under study corresponds to *Target B* from sequence *acrobats* from the SMOT dataset. It is worth noting that with our approach, the target position is remarkably precise, while if only $SiamMask$ was used, the target was almost lost around frame 60 and definately lost at frame 90, as it was shown in *Figure 1.2.1*. If we focus on the top-right graph, corresponding to the position signals of the vertical component of the centroid, we distinguish four different signals: the Ground Truth appears in relative transparent circles, in a more intense green is the proposed position by the appearance-based tracker, if the DM has determined that predictions must be made at that frame (vertical yellow lines), it appears as yellow circles and finally, the corrected position corresponding to the closest candidate is depicted in grey.

To understand the role that each of these signals play, let us focus around frame 55 in that same graph. At that instant, the siamese tracker proposed an incorrect bounding box, whose centroid was extremely inconsistent with the previous underlying dynamics. Therefore, the DN predicts a new sample based on the past trajectory, which is much more coherent. That new position is used to find the closest bounding box candidate, which is fed again into the tracker. Sometimes, the sequence is so challenging (such as occlusions or intersection of identical objects) that it takes the tracker some frames to recover the correct target, and for that to finally happen, it is crucial that the DN consistently provides correct positions in the meantime.

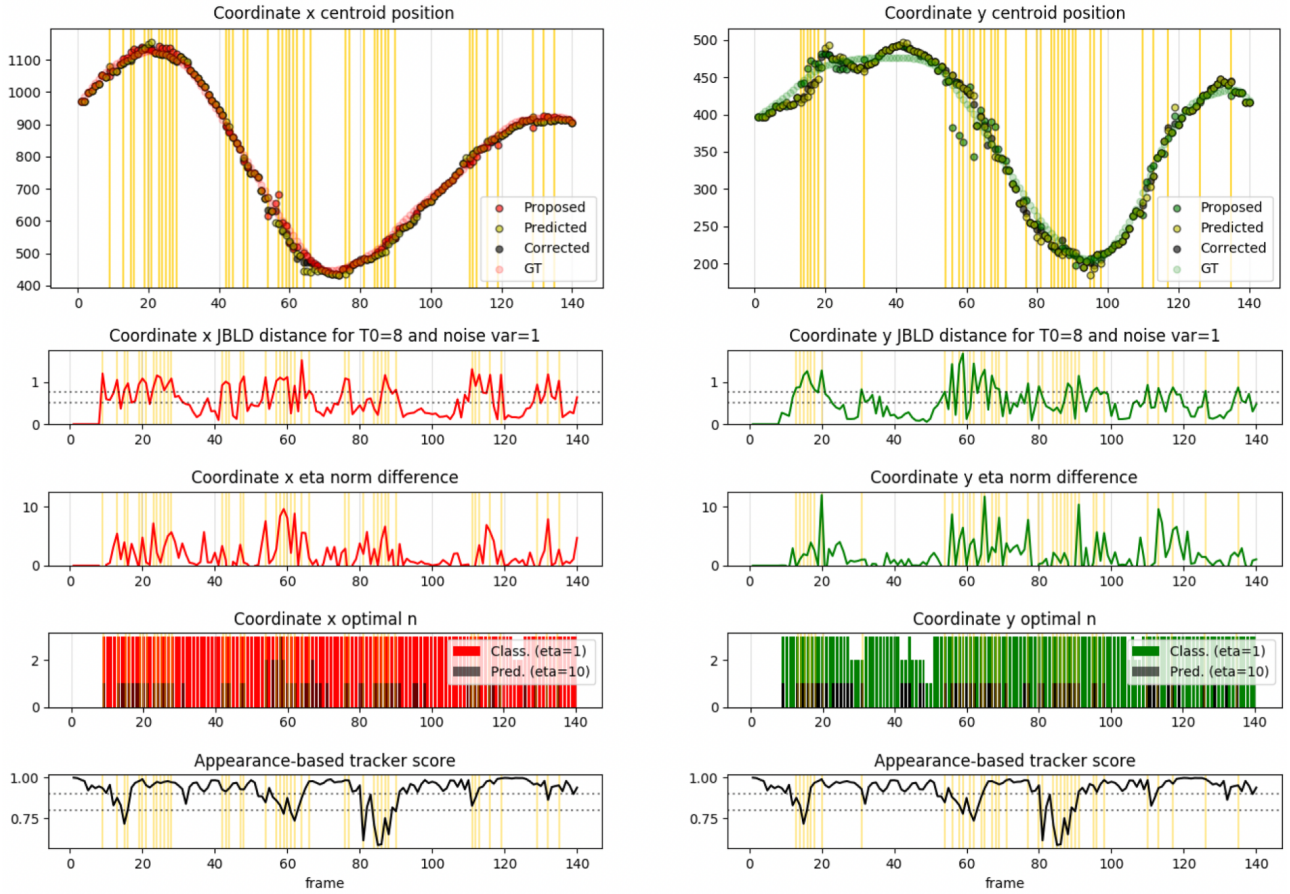

Figure 3.3.1: Set of graphs that exemplify the classification and prediction algorithms, as well as their interaction with the tracker. From top to bottom: 1) Each centroid's position in the Ground Truth (GT), the proposed position by the tracker, the predicted by the algorithm and the position of the chosen bounding box candidate. 2) JBLD distances for each coordinate. 3) η difference between $||\hat{\eta}_k^{(2)}||_F - ||\hat{\eta}_k^{(1)}||_F$ for each coordinate. 4) Optimal estimated order n for classification and prediction and for each coordinate. 5) Score provided by the visual tracker at each time frame (shared by coordinates). Vertical yellow lines represent where the Dynamics Module has determined that it must use prediction and horizontal lines represent thresholds.

Chapter 4

Experiments

In this chapter, we detail the implementation features of our work, as well a presenting the datasets and metrics we have used to evaluate our proposed approach. Additionally, we provide a variety of results, both qualitative and quantitative, for single and multiple object tracking.

4.1 Implementation Details

This thesis combines a *state of the art* computer vision network with a dynamics theory algorithm that comes from a robust control background. Although these two environments are closely related and its combination has proven to be effective, the current trend is to work on different languages for each of them. This work statrs from the official Python/PyTorch implementation[1] of [21], with a MATLAB implementation of Dicle's work in [5], shared by the authors.

We begun our research implementing our proposed Dynamics Module in MATLAB, leveraging the code provided by Dicle *et. al*, to check the feasibility of the dynamics theory behind our approach. Later on, in order to integrate our module with the tracker, we re-implemented from scratch all the DM in PyTorch, included Dicle's algorithms and we built the rest of our architecture on top of that. The implementation for our work, as well as interesting qualitative results of our approach in *GIF* format, can be found at our GitHub page [2].

The initialization file that is fed to our architecture determines whether it will perform single or multi object tracking, depending on the number of objects that are initialized in it. The majority of the parameters that conform our work are static, however, two parameters can be fine-tuned in order to obtain better results for distinct situations, and its variations have an important impact on the speed of our approach. These parameters are the following ones: the memory of the Dynamics Module T_0 and the number of candidates extracted by the tracker N. The intuition behind the choice of these parameters is quite obvious in most of the scenarios: if the target being tracked follows a simple dynamics, a small number of T_0 will provide good results, however, if its motion is quite challenging,

[1] https://github.com/foolwood/SiamMask
[2] https://github.com/marinalpo/DynamicTracking

the DM will need a larger window in which correctly encapsulate the object's dynamics. On the other hand, if the background of a sequence is uncomplicated, our algorithm will work perfectly with a small number N. However, if the background of the sequence contains distractors, a higher number of extracted candidates will be required.

Throughout this chapter, when providing qualitative and quantitative results for our approach, we will specify which configuration (combination of parameters) has been used among the following three:

- **Configuration A:** $T_0 = 8$ and $N = 75$
- **Configuration B:** $T_0 = 11$ and $N = 50$
- **Configuration C:** $T_0 = 11$ and $N = 75$

4.2 Dataset

We evaluate our approach on the Similar Multi-Object Tracking [5], a dataset that consists of eight videos where there appear multiple instances of identical or very similar objects. Some of the most dynamically challenging sequences in this dataset are *acrobats*, where five acrobats dressed in the same way lineup in the air and get occluded several times or *juggling*, where a juggler adds artistic motions to 3 balls with alternating tricks.

We have chosen to work on this dataset because it tackles precisely the phenomenom we are trying to solve in this thesis for, in order to obtain a good performance, it is crucial to use motion cues at some point, as the objects' appearance is quite ambiguous.

Additionally, we meticulously collected an extra set of challenging videos with multiple targets with extremely similar appearance, mostly of sports events such us NHL's hockey or UCL's soccer. We have added these sequences to the SMOT dataset, which is converted this way into the extended Similar Multi Object Tracking (eSMOT). Due to our approach's excellent performance on these sequences (see *Figure 4.4.1)*, we have used our rotated bounding boxes predictions as labeled ground truth. For this reason, no quantitative results will be provided for these sequence, as it would be unfair. In the future, a similar work of producing semi-automatic labels could be done with the original SMOT dataset, for at this moment it only provides axis-alligned bounding boxes as ground truth.

4.3 Metrics

We report the performance of our approach using three different metrics for single-object tracking and use two additional metrics for multi-object tracking.

For SOT, we report the mean Euclidian Distance between centroids (mED), that measures how far from the ground truth are the provided predictions. We also report the mean Intersection over Union

(mIoU), which is the ratio between the area of overlap and the area of union between the ground truth's and the prediction's bounding boxes.

Both for SOT and MOT, we report the mean Precision (mP) at a given IoU threshold. This metric expresses the ratio of frames in which the boxes corresponding to the labels and the detections present a IoU higher than a certain level, that is, the percentage of frames at which the prediction is considered to be valid (a match).

For multiple object tracking, we incorporate two measures widely reported for this specific task: Multiple Object Tracking Precision (MOTP) and Multiple Object Tracking Accuracy (MOTA)[1]. MOTA metric at a given threshold is defined as:

$$MOTA = 1 - \frac{\sum_k (FN_k + FP_k + MM_k)}{\sum_k GT_k} \tag{4.1}$$

where FN_k, FP_k, MM_k and GT_k are false negatives (misses), false positives, mismatches and ground truth at frame k. MOTP is the mean IoU for all matches over all frames, averaged by the total number of matches made M_k. It shows the ability of the tracker to estimate precise object positions, independent of its skill at recognizing object configurations, keeping consistent trajectories, and so forth. It is defined in the following way:

$$MOTP = \frac{\sum_{k,i} IoU_k^i}{\sum_k M_k} \tag{4.2}$$

Finally, both for SOT and MOT we also report the performance speed as *frames per second* (fps), which indicates how many frames the tracker can process per second. This measure changes substantially when solving Multiple Object Tracking, as more than one object is being tracked at each frame.

4.4 Results

We provide qualitative and quantitative results for our approach, both for single and multiple VOT. In all the tables and figures, we compare our approach with *SiamMask*'s performance, which we have computed under the same conditions in order to report a fair comparison. In order to be able to better appreciate the temporal evolution of our results, we strongly recommend to visit our previously provided GitHub page, where *GIF* examples of these results are shown.

4.4.1 Single Object Tracking

For Single Object Tracking, we present qualitative results for some of the extended SMOT dataset sequences. *Figure 4.4.1* exemplifies the performance difference between *SiamMask* and our approach for

	mED (pixels)				mIOU (%)				mP@0.8 IOU (%)				Speed (fps)			
ID	SM	A	B	C	SM	A	B	C	SM	A	B	C	SM	A	B	C
1	328.68	20.31	**18.73**	23.03	14.48	**50.46**	50.23	48.20	28.79	**100.00**	**100.00**	**100.00**	**34.14**	0.78	0.39	0.36
2	218.04	71.66	**14.9**	216.83	17.85	25.18	**50.04**	21.19	32.00	48.00	**96.67**	41.33	**37.33**	0.77	0.39	0.33
3	149.88	**15.22**	98.05	294.39	17.20	**56.48**	18.73	19.61	28.46	**100.00**	32.31	34.62	**33.91**	0.76	0.41	0.4
4	144.52	283.26	197.08	**16.6**	25.22	19.21	23.03	**54.37**	41.38	33.08	37.93	**98.62**	**34.91**	0.77	0.41	0.36
5	103.28	**10.92**	11.27	11.02	23.20	53.87	52.47	**53.97**	35.33	98.67	**100.00**	98.00	**38.41**	0.85	0.42	0.41

Table 4.1: Quantitative results of *SiamMask* (SM) compared to three different configurations of our approach (A, B, C) for each object of the sequence *acrobats* of the SMOT dataset.

a shared configuration *(Configuration A)*. On these sequences and with this configuration, *SiamMask* runs at a mean speed of 13.81 fps, while our approach operates at 0.81 fps.

In *Table 4.1*, we provide a quantitative comparison of the performance of our work with **SiamMask** for three different parameter configurations in the previously mentioned challenging sequence *acrobats* of SMOT. Our approach manages to track each of the five objects to (or almost to) perfection with a specific configuration but a shared combination of parameters that corrects each of the acrobat's trajectory at the same time has not been found. However, it is worth mentioning that the work presented in our thesis almost never is outperformed by the original siamese tracker. This interesting point, which will be later on also confirmed by the multiple object examples, leads us to believe that our approach is almost always better.

4.4.2 Multiple Object Tracking

We tested our algorithm for multiple target tracking on each sequence of the SMOT dataset with the same configuration (*Configuration B*). In order to have homogeneous sequences in terms of length and number of targets, we have limited the length of each SMOT sequence to 150 frames and the maximum number of objects being tracked to 10. This does not affect the fairness of our comparison, as the provided baseline results are tested with the exact same limitations. Results are reported in *Table 4.2*, where it can be appreciated that our approach outperforms *SiamMask* in five out of eight sequences, and also provides a competitive performance on the rest of the videos. As an average, our approach enhances the precision metric of the baseline by approximately a 5%, which we consider to be a success, given that our contribution has no training stage, and its parameters were not fine-tuned for each particular sequence.

In *Figure 4.4.2* we present some qualitative results for the *acrobats* sequence, being its correspondent quantitative results detailed in the first row of *Table 4.2*. In that figure, it can be appreciated how, although presenting some failure cases, our architecture outperforms *SiamMask*, completely correcting the trajectory of three out of five acrobats, while the baseline failed in all of them. To conclude this chapter, in *Figure 4.4.3* we exemplify the comparison of the baseline with our approach, both in the task of Visual Object Tracking and Visual Object Segmentation, in sequence *athletes* from the eSMOT dataset.

Figure 4.4.1: *SiamMask* (in red) and our approach's performance (in green) on three different sequences of the eSMOT dataset, from left to right: *soccer*, *hockey* and *football*. Top images correspond to the initialization frames. Sequence *soccer* consists on different players that run from one side of the image field to the other and return, being occluded during an important part of the sequence. It is interesting noting how our approach performs consistently even when there is a dynamic change in the player (in the 3^{rd} row it is going right and in the 4^{th} row it has turned to the opposite direction), while *SiamMask* lost the target at the first occlusion. In sequences *hockey* and *football*, different players with extremely similar appearance cross throughout the scene, making it impossible for *SiamMask* to maintain their identities, while our approach easily keeps them.

	mP@0.8 IOU (%)		MOTA@0.8 IOU (%)		MOTP@0.8 IOU (%)		Speed (fps)	
Sequence	SM	Ours	SM	Ours	SM	Ours	SM	Ours
acrobats	67.75	**87.84**	33.95	**75.39**	**52.84**	52.21	**11.3**	0.8
juggling	**66.57**	52.96	**32.58**	3.70	**52.75**	50.00	**11.94**	0.13
slalom	**65.38**	63.72	**30.18**	27.3	**56.35**	53.15	**14.39**	0.25
seagulls	76.51	**90.50**	32.87	**50.65**	51.40	**51.94**	**1.21**	0.02
crowd	98.08	**98.89**	37.59	**38.23**	55.86	**55.87**	**1.1**	0.01
tud-crossing	**87.03**	86.90	**73.59**	73.34	66.59	**66.82**	**3.28**	0.06
tud-campus	87.68	**99.64**	74.91	**99.64**	65.39	**65.65**	**8.78**	0.07
balls	71.76	**75.29**	43.53	**50.59**	60.41	**61.17**	**16.07**	0.23

Table 4.2: Quantitative results of *SiamMask* (SM) and our approach using *Configuration B* for each sequence of the SMOT dataset.

Figure 4.4.2: *SiamMask* and our approach's performance on the left and right images respectively. Figures on the top correspond to frame 44 and figures on the bottom to frame 76 of the sequence *acrobats* from SMOT. The Ground Truth for each object is depicted in each image in a thin bounding box. On frame 76, all acrobats are heading towards the center of the image, where they will cross, and then return again to the original position. It can be observed, how *SiamMask* has already lost one target before the crossing, and it loses all of them after it. On the contrary, although our approach does not give a perfect result, it tracks correctly three of the five objects.

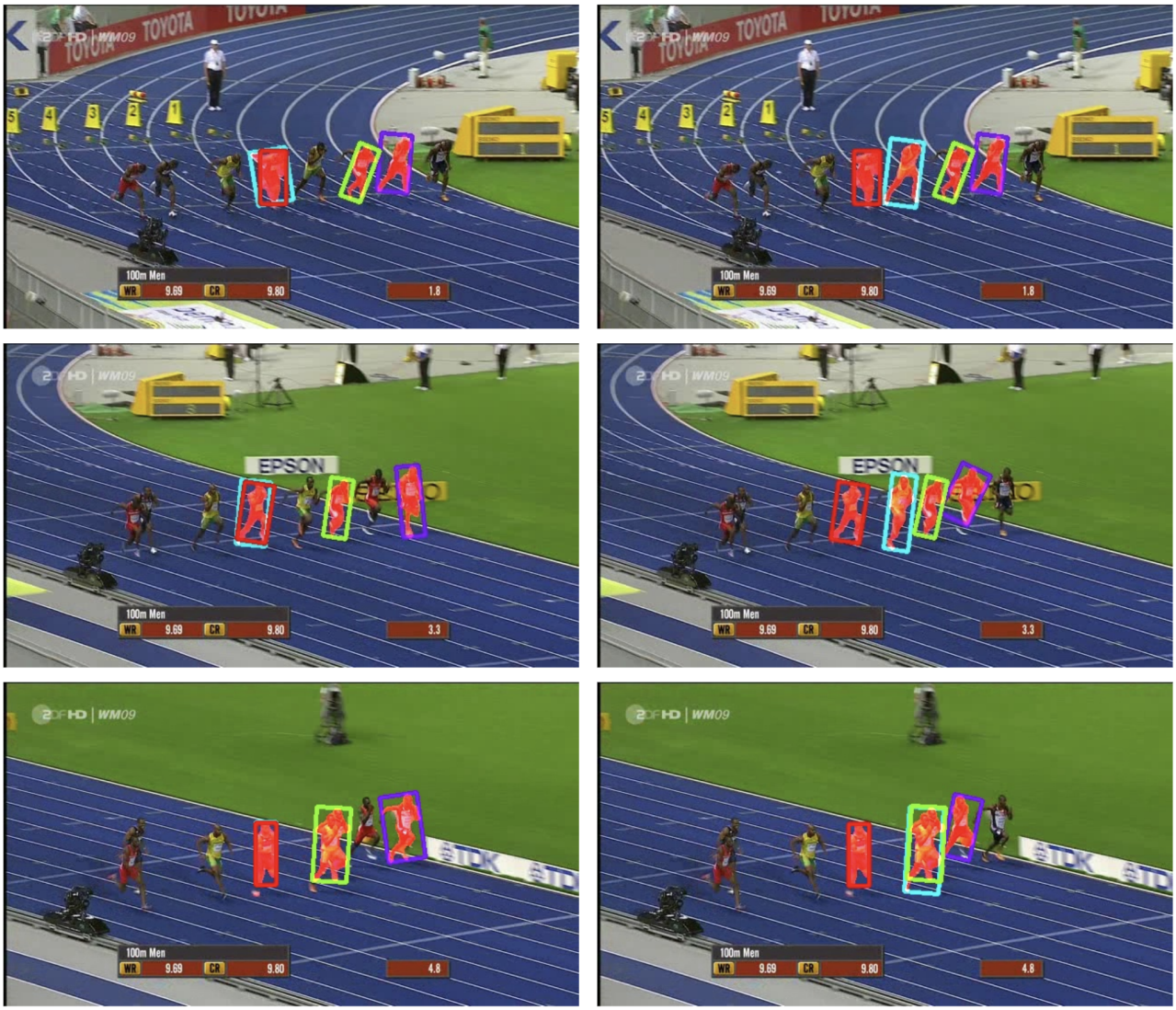

Figure 4.4.3: *SiamMask* and our approach's performance on the left and right images respectively, that correspond to some frames (decreasing from top to bottom) of sequence *athletes* from the eS-MOT dataset. These examples show how our model performs both on video object tracking and segmentation. No Ground Truth is depicted, for we have used our approach's predictions to label the sequences. It can be observed, how *SiamMask* loses two targets throughout the sequence, while our model maintains all of them in a quite precise manner.

Chapter 5

Conclusions and Future Development

5.1 Conclusions

In this work we tackled the task of visual object tracking in challenging scenarios, such as targets with identical or very similar appearance that interact between them. In these cases, we showed that appearance-based trackers fail to track the correct target and are easily fooled by scene distractors. We demonstrated that motion dynamics provide strong and useful cues to correct these tracker's detections by providing a novel and interesting way of combining *cutting edge* technology in the computer vision world with traditional concepts of control theory.

The main idea of our proposed approach is to, in an online manner, analyze the underlying dynamics of the incoming predictions proposed by an appearance-based tracker and determine if its motion is consistent with the past trajectory using a novel classification algorithm. We correct the provided sample if it is considered to be erroneous, and find the closest bounding box among a set of previously extracted and filtered alternative candidates.

We have carried out an intense study on the different configurations of our proposed Dynamics Module and presented several relevant insights about its composition, as well as illustrative figures to intuitively understand and analyze its performance. Additionally, we have inspected the behaviour of a *state of the art* deep visual object tracker, and performed the correspondent modifications successfully support a unified framework that integrated the Dynamics Module.

We illustrated the benefits of this framework with challenging qualitative results and showed that, when tested on the Similar Multi Object Tracking dataset, it outperformed the baseline's precision by an approximate 5%. We also introduced a novel extension to that dataset (eSMOT), that consists of extremely challenging sequences of sports events with multiple objects with similar appearance, and use our architecture to label the sequences.

5.2 Future Work

Given the results presented throughout this document, we assess that our proposed work emerges as a promising line of research for our Laboratory. However, there are several key improvements that should be done to the current architecture in order to transform it into a competitive tracker in the field.

We believe that the main problem with our proposed architecture is the computational efficiency, for it operates at a frame rate more than ten times slower than the baseline we are comparing to. Several things could be done to improve that aspect, being the following what we believe that should be done first.

Some improvement could be done in the current implementation of our approach. As it was conceived as a proof of concept, no extensive efforts were put on providing a computational efficient solution. However, once it has been proven that this is a valid approach, it would be worth trying to perform some implementation changes, leveraging mathematical concepts behind the control theory such as matrix sparsity.

Similarly, we believe that some slight changes could be done in our classification algorithm in order to avoid having to solve an Iterative Hankel Total Least Squares problem at each frame. A possible simple solution could be to implement some kind of pre-classification algorithm that using high level dynamic features such as pixel velocity or acceleration, determine if it was worth running the classification algorithm, with the IHTLS, at that frame.

We also believe that our approach should be evaluated in the most relevant datasets for the tasks tackled in this thesis, such as the VOT challenge [12] for single object tracking or the MOT challenge [13] or even the Trajectory Forecasting Dataset [17] for multiple object tracking in order to benchmark our proposed architecture.

Bibliography

[1] Keni Bernardin and Rainer Stiefelhagen. "Evaluating multiple object tracking performance: the CLEAR MOT metrics". In: *EURASIP Journal on Image and Video Processing* 2008 (2008), pp. 1–10.

[2] Luca Bertinetto et al. "Fully-convolutional siamese networks for object tracking". In: *European Conference on Computer Vision*. Springer. 2016, pp. 850–865.

[3] Longtao Chen et al. "Grid-based multi-object tracking with Siamese CNN based appearance edge and access region mechanism". In: *Multimedia Tools and Applications* (2019), pp. 1–19.

[4] Guilhem Chéron, Ivan Laptev, and Cordelia Schmid. "P-CNN: Pose-Based CNN Features for Action Recognition". In: *2015 IEEE International Conference on Computer Vision (ICCV)* (2015), pp. 3218–3226.

[5] C. Dicle, O. I. Camps, and M. Sznaier. "The Way They Move: Tracking Multiple Targets with Similar Appearance". In: *2013 IEEE International Conference on Computer Vision*. 2013, pp. 2304–2311.

[6] Matteo Dunnhofer et al. "Visual Tracking by Means of Deep Reinforcement Learning and an Expert Demonstrator". In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)* (2019), pp. 2290–2299.

[7] James Durbin and Siem Jan Koopman. *Time series analysis by state space methods*. Oxford university press, 2012.

[8] Masoud Faraki, Mehrtash T Harandi, and Fatih Porikli. "More about VLAD: A leap from Euclidean to Riemannian manifolds". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 4951–4960.

[9] Susanna Gladh et al. "Deep motion features for visual tracking". In: *2016 23rd International Conference on Pattern Recognition (ICPR)* (2016), pp. 1243–1248.

[10] Dongyan Guo et al. "SiamCAR: Siamese Fully Convolutional Classification and Regression for Visual Tracking". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[11] R. E. Kalman and R. S. Bucy. "New results in linear filtering and prediction theory". In: *TRANS. ASME, SER. D, J. BASIC ENG* (1961), p. 109.

[12] Matej Kristan et al. "A Novel Performance Evaluation Methodology for Single-Target Trackers". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.11 (Nov. 2016), pp. 2137–2155. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2016.2516982.

[13] L. Leal-Taixé et al. "MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking". In: *arXiv:1504.01942 [cs]* (Apr. 2015). arXiv: 1504.01942. URL: http://arxiv.org/abs/1504.01942.

[14] Binlong Li, Octavia I Camps, and Mario Sznaier. "Cross-view activity recognition using hankelets". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2012, pp. 1362–1369.

[15] Michael and Andrew Blake. *CONDENSATION-Conditional Density Propagation for Visual Tracking*. 1998.

[16] Haesun Park, Lei Zhang, and J Ben Rosen. "Low rank approximation of a Hankel matrix by structured total least norm". In: *BIT Numerical Mathematics* 39.4 (1999), pp. 757–779.

[17] Amir Sadeghian et al. "TrajNet: Towards a Benchmark for Human Trajectory Prediction". In: *arXiv preprint* (2018).

[18] Skipper Seabold and Josef Perktold. "statsmodels: Econometric and statistical modeling with python". In: *9th Python in Science Conference*. 2010.

[19] Bing Shuai et al. "Multi-Object Tracking with Siamese Track-RCNN". In: *arXiv:2004.07786* (2020).

[20] Arnold WM Smeulders et al. "Visual tracking: An experimental survey". In: *IEEE transactions on pattern analysis and machine intelligence* 36.7 (2013), pp. 1442–1468.

[21] Qiang Wang et al. "Fast Online Object Tracking and Segmentation: A Unifying Approach". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[22] Xikang Zhang et al. "Efficient temporal sequence comparison and classification using gram matrix embeddings on a riemannian manifold". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4498–4507.